PHYSICAL SIMULATION OF MUSCLES AND BONES

Emanuel Dima[1], Corina Dima[1], Dan Cristea[1,2]

[1] "Al. I. Cuza" University of Iaşi, Faculty of Computer Science
16, General Berthlot street, Iasi, Romania, E-mail: {dimae, gvrinceanu@info.uaic.ro}
[2] Romanian Academy – the Iasi branch, Institute for Computer Science
16, General Berthlot street, Iasi, Romania, E-mail: dcristea@info.uaic.ro

Abstract

This paper presents a model used to animate a 3D facial mesh. The animation is entirely based on a physical system, thus no artistic skills are required to animate a face through morphing or rigging. The facial expressions are the result of the physical interaction between the different components of the system. The focus is on creating structures to accurately represent the muscles and the bones. The physical engine is essentially a mass-spring system, but it also supports some other types of structures, namely pressure cells and rigid bodies.

Introduction

Creating realistic facial animations has always been a challenge in computer graphics. Traditionally, animating a human-like character is considered a difficult task, since it requires a lot of talent and modeling experience. We propose an anatomically based approach to the facial animation problem. This approach gives the animator the possibility to control the facial expressions of the character using the muscles defined in the system. The muscles are connected to each other, to the skin and to the skull. Their contraction induces the movement of the skin, thus creating facial expressions of the avatar.

The goal of the project is to create a three-dimensional model of a human face, able to convey emotions resembling in detail to human-like natural expressions to the users who interact with it. When created, an emotional language will permit the translation of the parameters of the system into generally known emotions.

The possible applications of such a system are countless: creating avatars for different interactive applications, that are able to read a text or carry a conversation; create 3D replicas of human beings and use them in teleconferencing (so only the audio signal and the captured parameters are sent through the network); create artificial tutors, that can teach lessons and show emotional feedback (nod their heads, gaze at the students, approve or disapprove their actions); medical applications on which to study the new look of expressions after facial or mandible surgeries and, of course, creating computer-generated characters that play a part in video games or movies.

This paper focuses on the mass-spring system that we used as a physical framework, and on the structure and dynamics of the muscles and the bones. It is organized as follows: first, we make a tour among related work then we describe the physical system. The next sections present the bone and muscle models and the last section gives the conclusions.

Related work

There are different approaches to creating facial animations. They vary in the means they employ for obtaining the animation and in the final result. The explanation for such a wide variety of animation techniques is simple: creating state-of-the-art facial animation is difficult and expensive, both from an artistic and computational point of view. Very complex animations, which take into consideration accurate simulation of different anatomical parts of the human head usually involve huge computational load and could be cost prohibitive. Therefore, most systems are focused on a specific goal and they include into the model only the minimum resources that allow them to fulfill the goals, thus minimizing costs.

The most widely used method for creating facial animations is the morphing method. It implies having a set of facial expressions of the model and a set of feature points. The animation is obtained by interpolating the positions of these points between one facial expression and the other.

There is a standard for defining the facial expressions, called FACS (Facial Action Coding System). Paul Eckman developed it in 1978 [1]. FACS proposes a mapping between different movements and contractions of the facial muscles and the name of the resulting facial expression. The state of several muscles at a certain point in time is labeled as an *action unit* (AU). Each of these action units corresponds to a visible change in the facial expression of the model. The original system included 46 AU. An example is AU 26, *lid tightener*, created with the help of the *Orbicularis Oris* and *Pars Palebralis* muscles.

Nedel and Thalmann [2] proposed a simulation system where the forces inside a muscle are described using the concept of an action line. An action line connects the origin of the muscle to its insertion point. The action lines follow the external shape of the muscle. To add volume to the muscle, the action lines are subsequently connected horizontally, using equally spaced ellipses. The resulting lattice is introduced into a mass-spring system, and particles are considered at the intersections between the action lines and the horizontal ellipses. To pre-

serve the volume of the muscle during the contraction some extra springs are added, called angular springs. For each pair of original springs, x_0x_1 and x_1x_2, two extra springs are added: one that connects the ends of the original springs, x_0x_2, and one that connects the particle x_1 with the middle of the x_0x_2 spring. The interior of the muscle in not modeled.

Another muscular model was the one used by Kähler [3]. In his approach, a muscle is represented as a bundle of fibers, each with a piecewise linear polygon as a control structure. Geometric shapes are attached to the segments of this control polygon. The fibers can form sheet muscles by grouping, while the union of the geometric shapes attached to the segments represents the muscle surface. This model supports two types of contraction, a linear muscle contraction and a sphincter muscle contraction.

The physical system

In order to simulate movement of the skin and the muscles in a physically correct manner, we need a system of objects that observe some clearly defined physical laws. Our choice was to implement a mass-spring system, that is, a system of mass **particles** connected by **springs** that observe Newton's laws of motion and Hooke's law of elasticity.

Each particle in the system has an identity, a position in a three-dimensional space (x, y, z), a mass (m) and a velocity (v). The fact that the particles have identity is not directly used, but it implies that two particles with the same position and mass are actually distinct. The mass of all muscle particles is the same, a parameter of the system; at the initialization of the system, these masses as well as their initial positions have to be specified. The positions of the particles are modified by the system during the animation, according to the laws of motion that take into account the velocity.

The springs (considered ideal) are characterized by an elasticity constant (k), a default length (l_r) and an actual length (l). Each spring connects exactly two particles. The elasticity constant and the default length are parameters of the spring; the actual length is computed as the distance between the two particles. A spring acts upon its particles with an elastic force proportional to the elasticity constant and the difference between its default and its actual length.

Although the model allows for a particle to be free (not connected to any spring), in our model all particles are bound to one or more springs. Our model also allows two particles to be connected by more than one spring.

The mass-spring system has an initial default state, as defined by the current position of all the particles. The system advances from one state to another according to the mathematical expressions of the physical laws that it emulates. The new state is usually computed using numerical integration methods.

We use the velocity Verlet method [4], as it provides a good trade-off between speed and accuracy. It is a second order integrator, usually used in molecular dynamics simulations, faster than the Runge-Kutta method [5] and more accurate than the Euler integrator [5]. Moreover, it simplifies the process of introducing constraints in the movement of the particles.

In our implementation, the advancement of the system from one state to another is called a *tick*. The actual *tick* is detailed in two steps: first, all the elastic forces, created by the springs, are computed. At the end of this step, for each particle, we know the net force of all the springs connected to that particle.

The second step consists of computing the actual movement of each particle, according to the Newton's equations of motion. The velocity of a particle is reduced by a percent representing the drag of the system. The acceleration is computed using Newton's second law, as the ratio between the net force that acts upon the particle and the mass of the particle.

In reality, to model realistically organic tissues more than simple mass-springs are necessary. For instance, when a muscle is contracted, its volume remains approximately constant. Also, bones are rigid objects, and therefore forces applied to them do not result in their deformation.

The muscle bulging behavior is simulated by the pressure cell model. A pressure cell is the equivalent of a surface that encloses a volume filled with ideal gas [7]. We use this to simulate the behavior of an anatomical cell, which maintains its volume disregarding the form that it takes. A gas-filled pressure cell does not constrain the volume to be perfectly constant, but is a good approximation of it.

The surface of a pressure cell is represented by a triangular tessellation, where the vertices are represented by particles. This surface should be closed as the volume of the pressure cell is computed based on it.

The implementation uses the ideal gas law, expressed by the equation $PV = nRT$. P is the absolute pressure of the gas, V is the volume of the gas, n is the number of moles of gas, R is the universal gas constant, and T is the absolute temperature. Because we are simulating an isothermal process, it means that we can compute the pressure inside a cell like this: $P = P_0 V_0 / V$, where P_0 is the pressure in the cell in the initial state (specified by the user), V_0 is the initial volume and V is the volume of the cell in the actual state.

Once we know the pressure, we can easily compute the force that acts upon the particles, using the formula: $P = F / A$ (Pressure P is the force F over the surface area, A).

The bone movements are those of a rigid body. A rigid body consists of a collection of particles that maintains its shape under rotation and translation. The actual implementation computes the torque (τ - the rotational force) generated by the forces that act upon every particle in the rigid body. The torque generates a change in the angular momentum (L) of the rigid, which is proportional to the angular velocity ω. The formula that we use to find ω is $L = I\,\omega$, where I is the moment of inertia and can be computed from the masses of the individual particles [6].

Bone structure

The physical structure of the bone is based on a three-dimensional object created in a modeling application. The object is a mesh, a triangular tessellation of the surface of the bone. For each vertex of this mesh we create a particle with a specified mass and the position of the vertex. All these particles define a rigid body. A rigid body does not allow individual movements of its particles. Applying a force upon a particle of the rigid body moves the whole rigid body. The distances between the particles that pertain to the rigid body do not change when they interact with the other components of the system. This allows us to move the bone without deforming its visual appearance, similarly to the way the bones move in reality.

The insertions that connect a bone with a muscle are springs, connecting particles from the rigid body with the particles that define the muscle. When a muscle is contracted, it acts upon the springs in the insertion and indirectly upon the bone, inducing its movement. For example, an action like opening the jaw can be achieved by contracting the platysma muscle under the chin. Since this muscle connects the jaw and the top of the spine, its contraction induces a rotation of the jaw in the vertical plane (Figure 1).

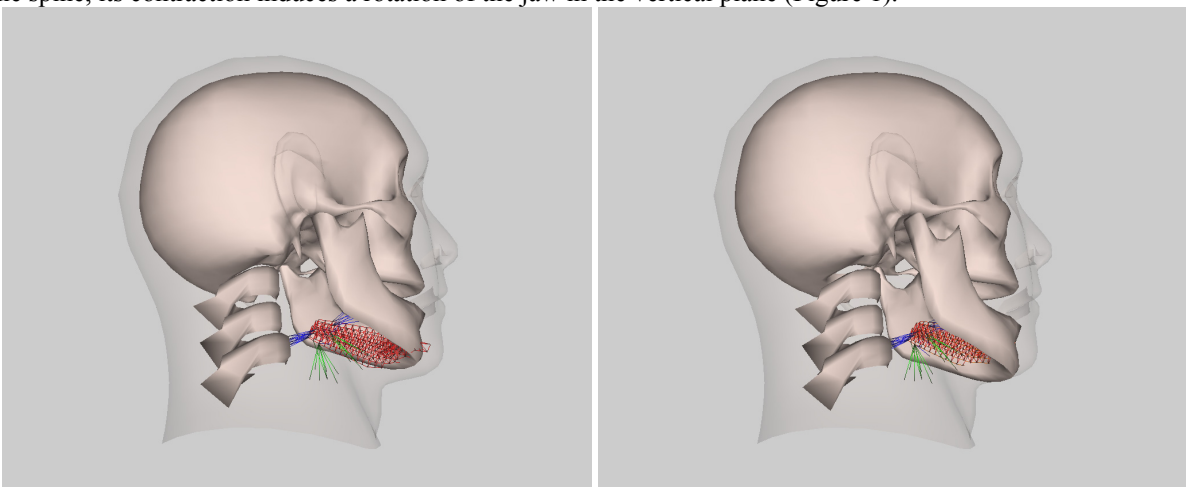

Fig.1. Opening the jaw by contracting the platysma

Muscle structure

Our muscle structure is based on two three-dimensional objects, created with a commercial modeling application. The first one represents the surface of the muscle, in our case manually modeled based on anatomical references. The second object is a three-dimensional poly-line that represents the contraction axis of the muscle, specified by a human modeler. The muscle structure consists of a three-dimensional particle lattice (particles connected by springs) that fills the inside of the muscle, having a featured subset of springs that are oriented on the direction of the muscle contraction axis.

The generation process of this lattice starts from the poly-line. We consider several equally spaced reference points on it. Each such point is contained in a plane that is perpendicular to the contraction axis. This plane intersects the surface of the muscle, thus generating a planar polygon. The inside of this polygon represents a muscle cross section that is perpendicular to the contraction axis in the corresponding reference point.

We then consider the cross section with maximum area as the main cross section. This cross section is filled with a planar lattice of mass particles and springs, actually a tessellation made of equilateral triangles with the area specified at the initialization. For each of the remaining cross sections, possibly having a different form than the main one, we build an analogous lattice through a bi-dimensional transformation, keeping constant the number of particles. Currently this transformation is based on radial similarity around the center of mass of the cross section: this means that the ratio between the distance from this center to a particle location and the distance from the center to the polygon outline is constant over all cross sections. Obviously, the lattices from the secondary cross sections are not regular anymore: the triangles become distorted.

The final step consists in connecting all the planar lattices through springs. Each cross section has the same number of particles. Longitudinal springs link two corresponding particles located on consecutive cross sections. By connecting with springs two triangles of consecutive cross sections a polyhedron is obtained. For

each one of these prisms we create a pressure cell in the physical system, which can be considered as the equivalent of a muscular cell. Our muscle now consists of a set of cells, which helps keep its volume approximately constant and generates internal resistance the same way real cells do. A contraction of the muscle, always taking place on the direction of the poly-line axis, shortens the longitudinal dimension of the polyhedrons and enlarges the triangles that form their bases. This determines the whole muscle to expand on the direction that is perpendicular on the contraction, and enhances the realism of its movement (Figure 2).

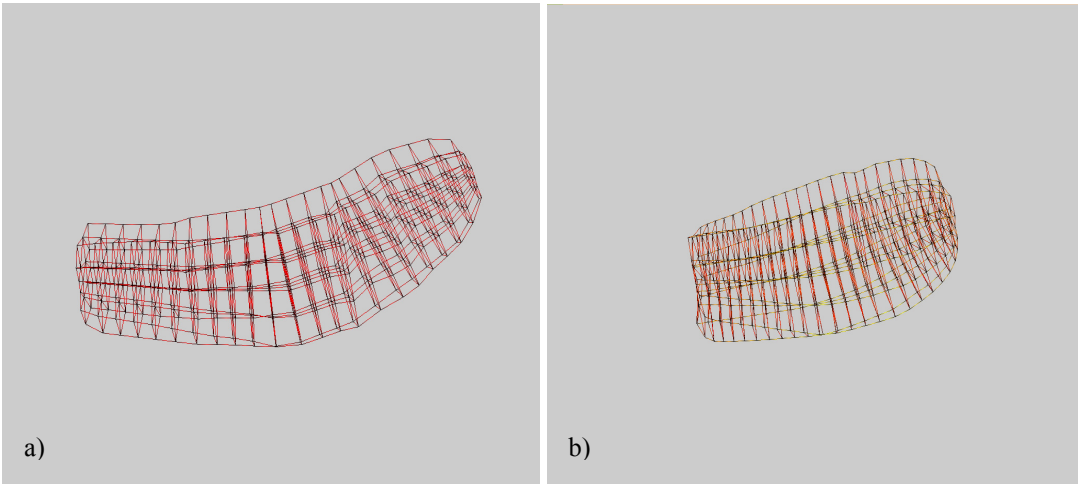

Fig.2. a) Relaxed muscle; b) Contracted muscle

Conclusions and future work

We presented a physical engine suitable for simulating anatomical entities. The engine is a mass-spring particle system, with rigid bodies and pressure cells simulation capabilities. The anatomical entities that we focused on are the bones, modeled by rigid bodies; the muscles, modeled by springs and pressure cells, and the skin and the insertions, modeled by springs only.

One of the problems in our simulations was the intersection between all these entities. When a muscle moves it should bulge under it, not start entering the deepest layer of the skin. In the current physical system, the muscles present such an undesirable behavior. The engine can be improved by adding collision detection and response capabilities. Thus, some important details of the human mimics would become easy to simulate.

References:

[1] Ekman Paul, Friesen V.Wallace, Hager C. Joseph, Facial Action Coding System Investigator's Guide. A Human Face, 2002.
[2] Nedel Porcher Luciana, Thalmann Daniel, Real Time Muscle Deformations Using Mass-Spring Systems. Swiss Federal Institute of Technology. Proceedings of the Computer Graphics International, 1998.
[3] Kähler, K., "A Head Model with Anatomical Structure for Facial Modeling and Animation", Master's Thesis, http://www.gamasutra.com/education/theses/20050526/Dissertation_Kaehler.pdf
[4] Ercolessi, Furio, "The Verlet Algorithm", A molecular dynamics primer, http://www.fisica.uniud.it/~ercolessi/md/md/node21.html
[5] Press, William H., Teukolsky, Saul A., Vetterling, William T., Flannery, Brian P., chap. Integration of Ordinary Differential Equations in Numerical Recipes in C. The Art of Scientific Computing, second edition, Cambridge University Press, 1992, pp. 710.
[6] Eberly, David H., Game Physics, Elsevier, 2004.
[7] Matyka Maciej, Ollila Mark, Pressure Model of Soft Body Simulation, http://arxiv.org/pdf/physics/0407003